Appropriate Machine Learning Methods for Service Recommendation Based on Measured Consumer Experiences Within a Service Market

Jens Kirchner
Karlsruhe University of Applied Sciences
Linnaeus University
Email: Jens.Kirchner@hs-karlsruhe.de, Jens.Kirchner@lnu.se

Philipp Karg and Andreas Heberle
Karlsruhe University of Applied Sciences
Moltkestr. 30, 76133 Karlsruhe, Germany
Email: kaph1014@hs-karlsruhe.de, Andreas.Heberle@hs-karlsruhe.de

Welf Löwe
Linnaeus University
351 06 Växjö, Sweden
Email: Welf.Lowe@lnu.se

***Abstract*—The actual experience of the performance of services at consumers' side is a desirable foundation for service selection. Considering the knowledge of previous performance experiences from a consumer's perspective, a service broker can automatically select the best-fitting service out of a set of functionally similar services. In this paper, we present the evaluation of machine learning methods and frameworks which can be employed for service recommendation based on shared experiences of previous consumers. Implemented in a prototype, our approach considers a consumer's call context as well as its selection preferences (expressed in utility functions). The implementation of the framework aims at the time-critical optimisation of service consumption with focus on runtime aspects and scalability. Therefore, we evaluated and employed high-performance, online and large scale machine learning methods and frameworks. Considering the Internet as a service market with perpetual change, strategies for concept drift have to be found. The evaluation showed that with the current approach, the framework recommended the actual best-fit service instance in 70 % of the validation cases, while in 90 % of the cases, the best or second best-fit was recommended. Furthermore, within our approach employing the best method, we achieved 94.5 % of the overall maximum achievable utility value.**

***Keywords*–Service Selection; Service Recommendation; Machine Learning; Big Data.**

I. INTRODUCTION

Service-Oriented Computing (SOC), Software as a Service (SaaS), Cloud Computing and Mobile Computing indicate that the Internet develops itself to a market of services where service consumers can dynamically and ubiquitously consume functionality from services with limited or no knowledge about the implementation or the system environment of the provided service. Besides the functionality, service consumers are interested in the performance of a service, which is expressed in its non-functional properties (NFPs) such as response time, availability or monetary charges. Within a service market, service functionality may be provided by several competing service instances. Among these similar services, service consumers are interested in the consumption of the service instance which fits best towards their preferences. In [1], we described that service selection has to be based on the actual experience of NFPs at consumer side. We defined *Service Level Achievement* to be the general performance of a service at consumer side (consumer side measured NFPs). In the case of basing service selection on Service Level Achievements, a service broker can automatically select the best-fit service among functionally similar services. In contrast, when the selection is based on Service Level Agreements (SLAs), one can only state the fact that SLAs have not been met; mitigating the issue, however, requires human action, hence, time. Perpetual change is one of the major characteristics of service markets such as the Internet. NFPs of service instances can be volatile (e. g., high load at certain times and limited resources), new functional equivalent instances enter the market; others are temporary or permanently not available. A collaborative knowledge base of consumption experiences benefits single users and helps them to optimise service selection towards their needs and based on their call context. Call contexts consider aspects that have an influence on NFPs at consumer side such as location, time, weekday, etc. When recommendation is based on actual experienced NFPs, their preferred weight from consumer side and the call context, the knowledge base has to be built on a potentially high load of measurement data, which needs to be processed and learned promptly. Within our framework, we develop a service broker which bases its decisions on consumption measurements of previous service calls [1]. In particular, service recommendations are based on a call context and a user's preferences which are expressed in a utility function. Basing service selection on Service Level Achievements requires measuring the NFPs of services at the moment of the actual service call. E. g., measuring response time of a service at a point in time called from a certain location. This can be easily integrated in SOC/SaaS infrastructures. However, it also requires the aggregation of individually measured data to turn it into knowledge about the expected performance of services in the future. Machine learning is an obvious candidate for this aggregation. For this task, machine learning methods have to cope with a high load of data efficiently in real-time or short periods.

Machine learning cannot be used out of the box until we find answers to the following questions: 1) Which concrete machine learning algorithm can be applied effectively, i. e., with high accuracy in the prediction of achievements? 2) Are there differences in the accuracy of different algorithms in the prediction of achievements? 3) How can we apply effective machine learning efficiently, i. e., with a minimum impact on service (selection) time?

In this paper, we seek to answer 1) by selecting reasonable candidates and experimentally evaluating their prediction accuracy. Addressing 2), for the selected algorithms, we even assessed prediction accuracy and its effect on the actual utility of the consumers (here, response time and availability). Finally, in order to address 3), we defined an architecture that is split into a foreground and background model. The service recommendation knowledge in the foreground model is used for fast service selection. It is asynchronously updated with the knowledge output of the background model, which, in turn, is gained by (potentially not quite as fast) machine learning on service call measurements at consumer side.

For the evaluation of appropriate machine learning approaches and their implementations in machine learning frameworks which can be employed for best-fit service recommendation, our focus was set on speed, real-time/online processing, accuracy and concept drift mechanisms. “Concept drift occurs when the values of hidden variables change over time. That is, there is some unknown context for concept learning and when that context changes, the learned concept may no longer be valid and must be updated or relearned” [2]. The evaluation was conducted with a prototype implementation of a holistic recommendation component for a service broker in order to determine the most appropriate method and framework for this purpose.

In Section II, the architecture of our service selection framework is introduced. Section III describes the challenges and optimisation focus within service selection. Section IV focuses on the pre-selection and evaluation of machine learning methods and frameworks. In Section V, the conducted evaluation of machine learning methods regarding appropriateness for service selection in a service market is described. Related work is outlined in Section VI. Finally, the conclusion is done in Section VII.

II. Framework architecture

In [1], we introduced our framework which optimises service selection based on consumer experience, call context and preferences (utility functions). In this paper, we focus on the recommendation component of our framework, which is depicted in Figure 1. The illustration shows the service consumer’s realm where a local component manages dynamic bindings and requests for best-fit service instances. The second task is to measure a service call’s performance (measurement of experienced NFPs) and to provide this information to the central broker component for learning purpose. Within the scope of this paper, our main focus is set on the foreground and background model combination, which is important for the overall recommendation process.

When a measurement feedback ticket arrives, the collected data is persisted and the pre-processing of the data is conducted. Afterwards, the data is used for the online learning of the NFPs. There exists a learning model for each NFP and service instance combination. In time or amount intervals, for each utility function and call context, the utility values for each service instance are calculated. Once all utility values for all instances are calculated, the service instance with the highest utility value within each utility function and call context will be updated in the foreground table. This foreground table contains the recommendation information for each call context and preference (utility function). When a service recommendation request arrives, the broker only has to look up this information in the recommendation table of the foreground model, while the actual learning and calculation is done asynchronously in the background model.

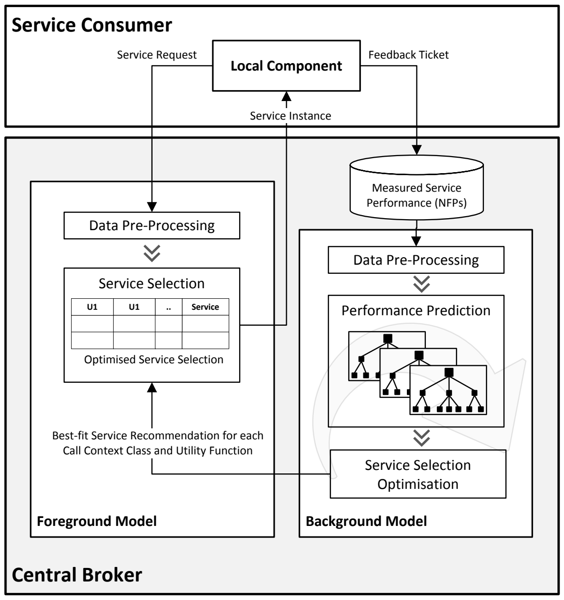

Figure 1. Foreground and background model within our framework.

The background model activities comprise machine learning, calculation and determination for the optimisation at consumer side. However, these tasks are time-consuming. The decoupling of the time-consuming tasks of the background model from the foreground model, which handles the time-critical recommendation tasks, reduces or even avoids the costs in terms of (service) time.

III. Optimisation focus and challenges

Following the Cross Industry Standard Process for Data Mining (CRISP-DM) [3], before machine learning methods can be employed, it is mandatory to have a clear understanding of the optimisation goals as well as the data which is used. As introduced above, the service broker and its learning component are supposed to recommend the best-fit service instance among functionally equivalent candidates. In particular, they are supposed to be aware of the call context and a consumer’s preferences and base their recommendation on the experiences of previous service calls by consumers of similar call contexts.

In general, the selection of a service instance is based on one or more NFPs. NFPs have different scales of measurement with different optimisation functions. For example, response time is a ratio scale with an optimisation towards the minimum. The availability for a service at a specific time is nominal: a service is either available or not. In such a case, the optimisation focus is to select a service instance which has the highest (maximum) probability of being available. When the selection of a service instance is based on more than one NFP, NFP data has to be normalised in order to be comparable and calculable. Usually, in such a case, not all NFPs are equally important, so their importance has to be weighted and taken into account. Within our framework, utility functions are used to calculate the utility value for each service instance based on the expected NFPs of each candidate and their weighted impact. “Utility functions can be captured in a vector of real values, each representing the weight of a corresponding quality metric [NFP]. The idea is based on a weighted scoring model”

[1]. For instance, lowest response time is more important (weighted: 60 %) than lowest price (weighted: 40 %) would results in a utility function $U(ResponseTime, Price) = 0.6 \times ||ResponseTime|| + 0.4 \times ||Price||$, where $|| \cdot ||$ normalises $ResponseTime$ and $Price$, respectively, between 0 and 1 [1]. Since preferences vary, the utility functions also vary among all service consumers. Therefore, within the overall optimisation/recommendation process, the overall utility is individual. However, within a single tier recommendation, the NFPs of service instances are based on consumers' contexts. Hence, within the same call context (e. g., time, weekday, location, type/size of input data), consumers with different preferences experience statistically similar NFPs, but the calculated utilities are different due to different utility functions. Within our framework, the expected NFPs of service instances are learned and each individual utility value is (pre-)calculated, which is then used for the actual individual service recommendation.

Analysing the market, change is the most important characteristic regarding the data. Besides new service instances, existing ones may temporarily not be available or cease to exist for good. But one has also to take into account that changes in infrastructure, network or market participants' behaviour arise, which also affect the NFPs within certain call contexts. These things are in general not evident to consumers, but have to be taken into account for recommendation tasks. Therefore, learning components for service brokers have to cope with rapid change. Service recommendation is in total a time-critical challenge. First, change has to be discovered quickly, and secondly, the recommendation query itself is supposed to be part of a service call and service time is one of the major optimisation goals. However, service recommendation is time-consuming, especially the learning and calculation part of it. Within our framework, we tackle this drawback by introducing two approaches. The first approach splits the component for direct dynamic service recommendation requests into a foreground and a background model. The learning of the expected NFPs is done in the background model, while its results together with the pre-calculated utility values, and therefore the best-fit service instances within each utility cluster and call context class, are stored in the foreground model, which can be easily and quickly retrieved. The second approach focuses on dynamic service binding, which is mainly based on the idea that service bindings are updated at consumers' side. For this, we developed a plug-in for middleware/SOA products. The actual service binding addresses within the dynamic service binding are updated by the central component by the usage of the publish–subscribe pattern [1].

We predefined the following optimisation goals for the recommendation unit within our framework:

1) High-performance service determination
2) High recommendation accuracy of the best-fit service instance
3) Continuous machine learning process
4) Adaptation to performance (NFP) changes

IV. Machine learning methods and frameworks

This section focuses on the pre-selection and evaluation of machine learning methods and frameworks.

A. *Pre-selection of machine learning methods*

Based on [4][5], there are several aspects for the evaluation of machine learning methods such as speed, accuracy, scalability, robustness and interpretability. The requirements listed in Table I were defined for the pre-selection of appropriate machine learning methods.

TABLE I. REQUIREMENTS FOR THE SELECTION OF MACHINE LEARNING METHODS

Speed	describes how efficient the machine learning method performs concerning the training and prediction time. Furthermore, this aspect also concerns the overall machine learning process as a 'critical path' from end-user side.
Accuracy	describes how effective the machine learning method performs: Degree of correct classification or coefficient of determination in regression [5].
Scalability	considers the ability of the method to be efficiently applied to a large data set [5].
Robustness	describes the ability to make correct classifications and predictions, given noisy or missing data value. It also considers whether the method is able to run automatically in a changing environment [5].

There is a broad variety of learning methods that address classification and regression. The major aim of our recommendation unit is to recommend the best-fit service. Although the prediction of a certain value can be employed for pre-calculation purpose, the goal within the overall recommendation process is to recommend best-fit service instances, which is a nominal result. For classification in general, the costs of the learning phase are cheaper than for regression. Therefore, we focused primarily on classification methods in the initial phase. In [4], the author published a comprehensive overview of established supervised machine learning classification techniques. This overview provides useful information for method selection, highlighting the benefits and drawbacks of each method which helped us to find appropriate methods for further evaluation. Table II is an extraction reduced to the requirements we outlined in Table I.

As previously mentioned, although in general *accuracy* is an important desire in machine learning, in our time-critical domain, *speed* is as important within our recommendation process. However, due to the separation of the foreground and background model, speed is not as important anymore, since learning is done asynchronously in the background model. Therefore, we are able to put a stronger focus on accuracy again. *Scalability* and *robustness* are also important criteria in our framework. Although Naïve Bayes and Decision Trees are not highly rated on accuracy in Table II, their rating in all other criteria is high and, among all criteria, they have the highest overall rating. After initial pre-tests with test data sets of various kinds, we selected Naïve Bayes, Hoeffding Tree [6] and Fast Incremental Model Trees with Drift Detection (FIMT-DD) [7] for the implementation in our framework, and therefore, for the evaluation within our broker scenario.

The Naïve Bayes classifier is a simple probabilistic classifier based on Bayesian statistics (Bayes' theorem) with strong independence assumptions [8]. The Hoeffding tree or Very Fast Decision Tree (VFDT) is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples do not change over time. It exploits the fact that a small sample can often be enough to choose an optimal

TABLE II. COMPARISON OF THE MAJOR MACHINE LEARNING METHODS
(**** REPRESENT THE BEST AND * THE WORST PERFORMANCE) (BASED ON [4])

	Decision Tree	Naïve Bayes	Neural Networks	kNN	SVM	Rule Learners
Speed of Learning/Training	***	****	*	****	*	**
Speed of Classification/Prediction	****	****	****	*	****	****
Scalability / Incremental Learning	**	****	***	****	**	*
Accuracy	**	*	***	**	****	**
Robustness/Tolerance to missing values	***	****	*	*	**	**
Robustness/Tolerance to noise	**	***	**	*	**	*

splitting attribute. This idea is supported mathematically by the Hoeffding bound, which quantifies the number of observations needed to estimate some statistics within a prescribed precision [6][9]. FIMT-DD focus on time-changing data streams with explicit drift detection [7].

B. *Machine learning frameworks*

Besides the machine learning methods, their implementation within libraries and frameworks are as important. For the selection of machine learning frameworks, we considered the requirements listed in Table III to be relevant.

TABLE III. REQUIREMENTS FOR THE SELECTION OF MACHINE LEARNING FRAMEWORKS/LIBRARIES

Integration	Easy integration of the library in Java; Capability of online processing within a machine learning workflow (online learning)
Automation	High degree of automation; Ability to adapt to changes (e. g., changing service instances, service consumers with new call contexts)
Usage	Strong dependency between the library and the machine learning methods; Framework targets general approaches and is not limited to specific purposes
Open Source	Framework should be open source and freely available to the public

In order to get an overview about popular and wide-spread software in the data analytic and data mining area, we took the results of KDnuggets' online poll about software that their open community members had used within the past 12 months in real projects [10]. The results of the poll, in which 3,000 had participated, contain open source as well as commercial products. Although these results give an overview about the software in this field, it has to be considered critically, since the poll was open and everyone could have taken part.

The following frameworks were pre-evaluated based on our requirements listed in Table III, but were not suitable for our purpose. RapidMiner [11] is a well-known and widely-used desktop application for solving a variety of machine learning, data and text mining tasks. It offers a comprehensive selection of machine learning methods and integrates third-party libraries. Despite of all the benefits, we declined RapidMiner mainly because of the missing capability of incremental/online learning. A reason for that could be that RapidMiner comes from the area of classical batch processing and analytics. Furthermore, an integration in Java is possible but requires additional efforts for automation tasks. R [12] is an open source software environment for statistical computing and graphics. With a classical statistical background, R does not provide modern machine learning approaches and does not focus on incremental learning. Apache Mahout [13] is a suite of machine learning libraries with algorithms for clustering, classification and collaborative filtering on top of scalable and distributed systems. Despite the overall advantages, it was not selected because of the little selection of machine learning methods and the specific use cases. Other libraries such as Apache Spark [14], KNIME [15], Shogun [16], Shark [17], scikit-learn [18] and Vowpal Wabbit [19] were reviewed but not considered for further evaluation, because of early misfits to our requirements.

We selected Weka [20] and MOA [21] because of their extensive collection of classical machine learning methods as well as new algorithms with state of the art concepts for incremental learning. Because of their native Java integration ability, they provide a high degree of automation. Furthermore, Weka is also used by other software in this sector, such as RapidMiner. Both frameworks are open source and developed by the University of Waikato. Weka contains different methods and algorithms for pre-processing, classification, regression, clustering, association rules and visualisation. MOA stands for Massive Online Analysis, which focuses on online algorithms for data streams. It includes several state of the art algorithms for classification, regression and clustering.

V. EVALUATION

This section describes the evaluation scenario, evaluation criteria, the evaluation environment and method as well as the results of the evaluation of machine learning methods regarding their appropriateness for service selection in a service market.

A. *Evaluation scenario*

The evaluation of the learning methods and frameworks implies that the actual best-fit service instance is known at each call context (location, weekday, time, etc.) with each utility function. This is a challenge when it comes to a real-world validation. In reality, service calls in a service market, especially the Internet, cannot be repeated under the exactly identical conditions as the Internet's network behaviour as well as a service's infrastructure are complex systems in terms of factors that influence service calls. For instance, at a certain, unique moment, the load of a service instance's system environment and the network load or any incident are combinations of coincidences and can therefore not be repeated. In order to gain exact reproducible situations, all service calls which are supposed to be compared need to be made at the same moment which is practically infeasible, especially when there are several competitive service instances.

To get a situation where the validation process retrieves exactly the best-fit service instance for validation at each moment considering call context and utility function, we developed a simulator which creates service instance measurements for a certain time period based on predefined behaviour profiles. The implementation of this framework follows a periodic behaviour influenced by statistical random-based deviation. Currently, the periodic behaviour of the simulated Web services follows our initial measurements in [1] and considers: day/night time, weeks, months, work days and weekends. The random-based deviation is supposed to simulate unexpected incidences such as network traffic jams, high/low usage of a service's limited infrastructure. The random-based influence over a period was also evidenced in our real-world service tests [1]. At the moment, two NFPs are simulated which are response time and availability.

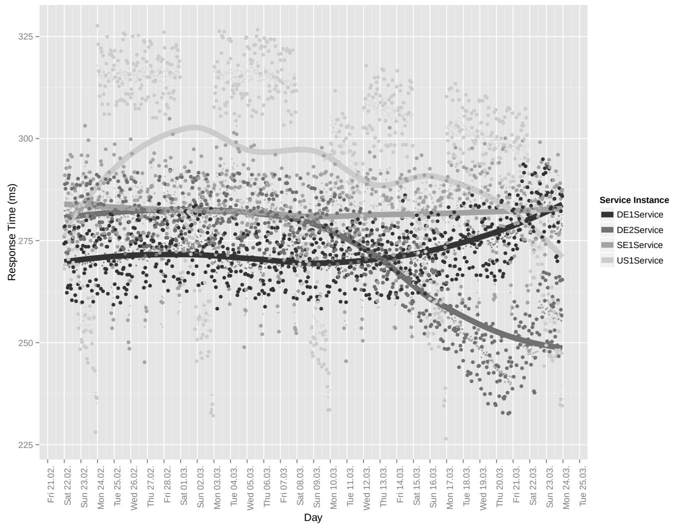

Figure 2. Overview about the simulated response time of four service instances and their trend over the whole period.

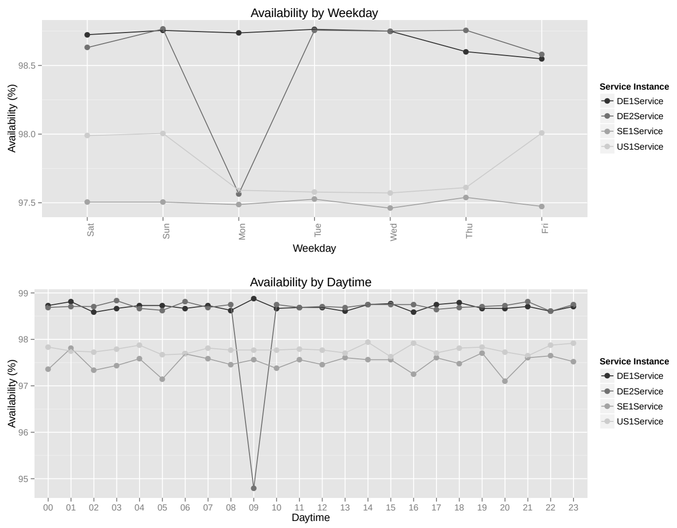

Figure 3. Overall periodic behaviour regarding availability of the simulated service instances with weekday and daytime aspects.

Figure 2 and Figure 3 depict an overview about the simulated NFPs. The simulated validation data set comprises a period of 30 days and has a total set of 460,800 records (40 records/hour $\times$ 24 hours/day $\times$ 30 day $\times$ 16 unique clients). The records contain information about day, time, response time in millisecond and availability (Boolean). Within the simulation, between each record there is a time interval of 90 seconds. Figure 2 shows in a condensed form the response time of all services instances within the whole period. Note that the line is only the trend. Within the recommendation process, the actual best-fit service instance at each time is important and not the averaged value of each service instance. The line is therefore only a visual orientation for us to determine the concept drift of each service instance within the period (e. g., DE2Service). Figure 3 shows the statistical value of availability with a focus on weekday and daytime periods.

B. Evaluation criteria

For the evaluation of the appropriateness of machine learning algorithms and their implementation in frameworks, each framework has to recommend the best-fit service instance based on a user's utility function and call context, i. e., the instance with the highest utility value. Implemented in our foreground-/background-model scenario, the machine learning algorithms have to estimate the expected NFP behaviour for all NFPs and all service instances. As described above, this results in $n \times m$ models (while n is the amount of considered NFPs and m is the amount of service instances). Since the number of modes can increase significantly, the recommendation table of the foreground model is updated asynchronously by the background model. In the background model, NFPs are learned continuously using the incremental learning functionality of the machine learning methods. However, although the expected NFPs per service instance are learned online, the utility values for each utility function and call context, and therefore the determination of the best-fit service instance, are calculated asynchronously in time- and count-based intervals. Besides the fact whether the actual best-fit service instance was recommended at each time, we also focus on how good the pre-calculated utility value was. For this, we took two indicators: *TOP1 accuracy* gives evidence on the recommendation quality as percentage of the correctly determined best-fit services with respect to the actual best-fit services instances, whereas the *TOP2 accuracy* shows the percentage of the determined best-fit instance within the actual top-two best-fit service instances.

C. Evaluation environment and method

For the execution of the evaluation, we used a test machine running Linux (Ubuntu 12.04 LTS) as its operating system, equipped with an i5-3340M CPU @ 2.70GHz x 4 (64 bit) and 15.6 GiB RAM.

The evaluation focuses on the overall recommendation. Recall that the overall process is split into the learning of the actual expected NFPs in a certain call context, for which we employ machine learning frameworks, and the calculation of the utility values according to service consumers' individual utility functions (preferences), from which the best-fit service can be determined. The machine learning frameworks learn each incoming record online, while the pre-calculation and determination part is conducted in intervals.

The evaluation is supposed to evaluate the accuracy within the period of examination at each point in time. Therefore, the data is not split into a training and validation set. In fact, with each learning interval, which also contains the update of the foreground table, the recommendation entry for a call context and utility function is validated with the actual best-fit service for the upcoming service call. Hence, at each time, with the previously learned records, the recommendation quality for future recommendation requests in the same call context and with the same utility function is measured. The idea is that change in general can be evaluated.

D. Evaluation results

As written above, the overall learning for the recommendation of the best-fit service instance is split into the learning of the expected NFPs, for which we employ the machine learning methods/frameworks, and the determination of the best-fit instance based on pre-calculation. The first part is done continuously, while the second part is done in intervals. For the overall evaluation, we conducted two rounds, one with an interval of 10 records and one with 100 records. Table IV shows the results for 100 records, since there were only minor differences between both rounds; however, the 10 records round achieved slightly better results. As shown, the results of the FIMT-DD achieved around 70 % of correct predictions (with 10 record intervals, we achieved over 70 %). Note that the calculated utility ranges from 0–100. Comparing all methods, there is not much difference between Naïve Bayes and Hoeffding Tree. The FIMT-DD shows very good results. It has the highest update rate of the foreground table, which is an indication that it reacts quicker and more fine-grained on change than the other methods.

TABLE IV. EVALUATION RESULTS OF THE MACHINE LEARNING METHODS NAÏVE BAYES, HOEFFDING TREE AND FIMT-DD WITHIN THE OPTIMISED SERVICE SELECTION/RECOMMENDATION

	Naïve Bayes	**Hoeffding Tree**	**FIMT-DD**
TOP1 Accuracy (in %)	58.634	59.837	69.287
TOP2 Accuracy (in %)	90.163	90.421	93.471
Mean Absolute Error (Utility)	1.656	1.660	1.049
Recommend. Table Updates	659	647	1.189

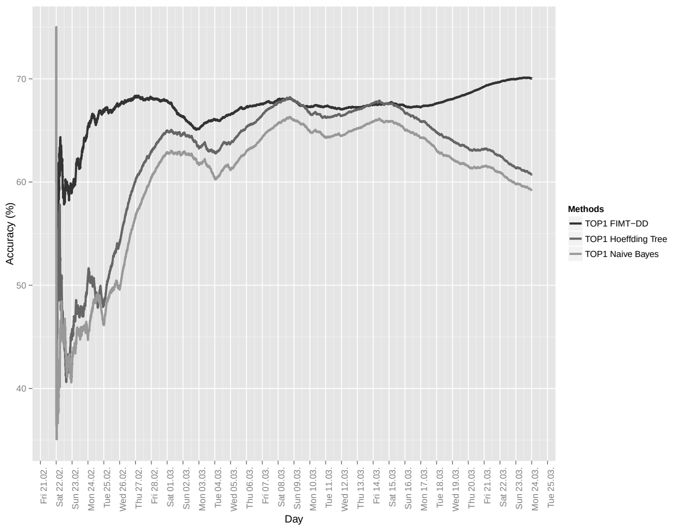

Figure 4. Service recommendation accuracy of the FIMT-DD, Hoeffding Tree and Naïve Bayes algorithm in the course of time.

The cold start problem applies to service recommendation, which means that good recommendation results are also supposed to be achieved with a small set of records at the beginning. Depicted in Figure 4, we can see that for the TOP 1 indicator in the overall recommendation process, the FIMT-DD quickly achieves a high accuracy. The drift detection of the FIMT-DD seems to work at the end of the period where some service instances change their performance behaviour (see Figure 2).

Our recommendation approach is supposed to be scalable. Table V shows the processing time of the overall process (NFP learning, utility pre-calculation, best-fit determination and update of the foreground table) for incoming measurement records. As we can see, a single record is processed in less than three milliseconds by a total of 460 thousand records. For all machine learning methods, we used the MOA framework. The figures show that, in term of learning overhead, there is not much difference between the methods. However, comparing the figures, Naïve Bayes would be able to process approximately 900 records more in an hour than the FIMT-DD. Comparing the numbers of the 100 record and the 10 record intervals (factor 8.8), it reveals that since the NFPs are learned continuously for every incoming record, the time-consuming part is related to the pre-calculation and best-fit determination. Although this is also due to the fact that these figures also include the evaluation steps (calculation of the actual NFPs, calculation of the actual utility values and determination of the actual best-fit instance), there is a high optimisation potential within the pre-calculation/-determination steps such as in-memory databases instead of hard disk databases. However, since these steps are asynchronous, they do not harm the recommendation process and are still good enough for background processing.

TABLE V. TIME FOR PROCESSING A SINGLE INSTANCE IN MILLISECONDS OF THE MACHINE LEARNING METHODS NAÏVE BAYES, HOEFFDING TREE AND FIMT-DD

Processing per record (ms)	**Naïve Bayes**	**Hoeffding Tree**	**FIMT-DD**
100 record intervals	2.602	2.614	2.621
10 record intervals	22.997	23.080	23.129

Figure 5 reveals more insight in the accuracy measure. The figure shows the degree of accuracy of the utility prediction. Once again, the best-fit service instance is the instance with the highest utility value regarding a service consumer's individual preferences (utility function). So, the closer the prediction towards the actual utility value is, the better the method. Comparing the method's charts, we see that for Naïve Bayes and Hoeffding Tree the predicted utility values at each time are both quite similar and do not reflect the curve of the actual values. In contrast, the chart for FIMT-DD depicts that the prediction is very close to the actual values. The intercepts of the curves show, that FIMT-DD does cope with change rapidly. In all cases, intercepts – which denote a change in the best-fit ranking – are also reflected in the prediction quite accurately.

For the evaluation of service recommendation in general, the actual utility gain is an important measure. Since the selection of service instances are based on several NFPs, the utility value as a basis for the individual preferences is an appropriate measure to benchmark service recommendation. In Table VI, the average experienced utility value after the service recommendation based on the FIMT-DD algorithm

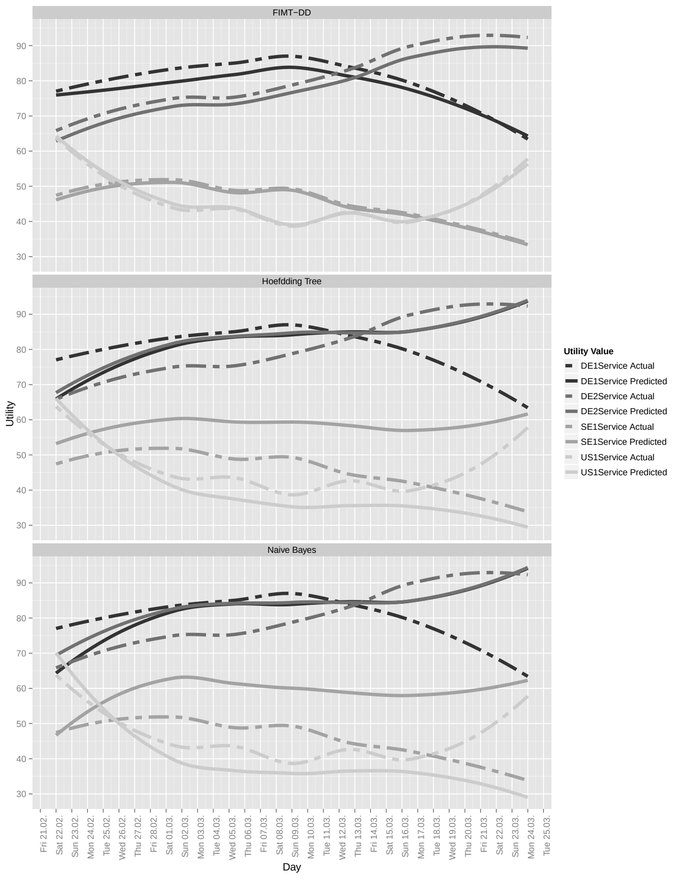

Figure 5. Detailed overview about the predicted and actual utility values.

TABLE VI. UTILITY GAIN WITH SERVICE RECOMMENDATION USING THE FIMT-DD ALGORITHM IN COMPARISON

After selecting ...	Average experienced utility value	FIMT-DD comparison in per cent
the FIMT-DD recommended instance	86.79	100.0 %
the perpetual best instance at each time	91.86	94.5 %
the perpetual worst instance at each time	29.22	297.0 %
the statistically best instance statically	81.96	105.9 %
an instance randomly	64.08	135.4 %

is compared with other scenarios. The table reveals good results. As written above, within this evaluation scenario, the overall best and worst services can be determined at each time. Once again, such comparisons are only possible within such a scenario; this is not possible in reality. Comparing the figures, we see that the FIMT-DD-based recommendation is able to achieve 94.5 % of the maximum achievable utility value. It is 35.4 % better than a random selection approach and even 5.9 % better than the statistically best service instance when statically using it. Note, that statically choosing the statistically best service instance is also a kind of learning.

VI. Related work

In [1], we introduced the overall concept of how knowledge can benefit service selection/recommendation in general. In that work, we presented the framework on an abstract level and introduced the recommendation component as a black box. In this work, however, we highlighted exactly this recommendation component and demonstrated the application of a concrete machine learning approach and how Service Level Achievements can be turned into knowledge for the benefit of a consumer-centric optimised service recommendation. In [1], we introduced a multi-stage selection with multi-stage dependencies of (compound) services. In this work, however, our focus was set on the general, appropriate and feasible approach of employing machine learning methods within service recommendation.

Further details about the evaluation can be read in [22]. This initial approach focuses mainly on the evaluation of machine learning algorithms which are appropriate for service recommendation. However, this work does not focus on some major characteristics of a service market such as the granularity of contexts and preferences (utility functions) and the availability of measurement data according to the actual usage of services. Furthermore, service recommendation also influences the behaviour of NFPs of service instances. For instance, best-fit service instances are more often consumed, which might affect response time. So, in case of limited service resources, these best-fit instances' NFPs can change for the worse.

Collaborative filtering (CF) approaches for service recommendation also focus on the exploitation of shared knowledge about services in order to recommend services to similar consumers before the actual consumption on an automated basis [23][24][25][26]. The major drawback of CF is that consumer-related preference similarities have to be found beforehand. With our call context and utility function approach, new consumers can already benefit from existing knowledge. CF approaches also do not take into account that consumers can have different optimisation goals or preferences and only some approaches [24][25] consider differences between consumers regarding their context. In [27], the authors tackle the lack of consideration of a consumer's preferences and interests; however, they do not take consumer context into account. The authors of [28] describe an approach to tackle the mentioned cold-start problem within CF.

Some work focuses on the prediction of NFPs for the detection of SLA violations such as [29]. This work mainly focuses on SLAs. In [1] (also cf. [26]), we argued that SLAs are not a good basis for service selection, considering the fact that service providers are profit-oriented, it is tempting to embellish their SLAs in order to be consumed. Also, as SLAs of consuming and providing services (e. g., compound services) depend on the SLAs of their sub-providers, deviations of actual non-functional characteristics and those specified in SLAs may propagate and spread even unintendedly and without the control of the providers. Furthermore, the performance experience at consumers' side is also dependent on a consumer's call context, which is also not reflected in SLAs.

We recently conducted a paper study about relevant NFPs during service selection, for which we processed over 4,000 conference papers in the SOC domain. In this study, we discovered only a very few papers considering more than one NFP during service selection/recommendation. Our approach considers the optimisation (recommendation) with several NFPs which is challenging due to the fact that the determination of the best-fit service instance according to service consumers' individual preferences result in a calculation task. Furthermore, NFPs have different scales of measurement and different optimisation focuses. Therefore, the complete recommendation process cannot be left to machine learning alone.

VII. Conclusion

We evaluated appropriate machine learning frameworks for the recommendation of best-fit services according to consumer preferences and call contexts within a service market. Implemented in the recommendation component for a service broker, the FIMT-DD showed very good results in the evaluation. Furthermore, its implementation in the MOA framework fulfilled a high degree of our requirements in terms of recommendation accuracy, speed of learning, scalability and robustness. With a high degree of automation, Java integration and being open source, the MOA framework also fulfilled all software-related requirements.

This initial evaluation was based on a continuously generated simulation data set. This data set allowed us to compare the learning/recommendation results with the overall optimum and on a reproducible basis, which would have not been possible with real-world services. The characteristics of the simulated data set, however, are based on real-world services, which we observed in former studies. The records in the data set consisted of continuous data. Within our framework and in reality, this continuity of records can in general not be assumed, since the recommendation framework uses measurement of actual invoked service calls. However, service instances cannot be assumed to be invoked equally often. Furthermore, best-fit services are more often recommended and therefore consumed, while underdogs never get the chance to prove themselves. Also, often recommended service instances might be affected by the over-consumption if infrastructure resources are exceeded. In order to tackle this, our framework needs further strategies and tests of how big the impact of this potential drawback is in reality and how this can be solved. The implemented prototype of the overall recommendation process also needs further performance improvements for the pre-calculation and pre-determination part. Finally, the overall approach has to be validated to a real-world scenario with a realistic number of clients and services.

Nonetheless, the prototype and the employment of the FIMT-DD within the MOA framework build a good foundation for service brokers recommending a best-fit service towards service consumers' preferences. With the foreground/background architecture of the framework, the time-consuming overall learning process is decoupled from the actual time-critical recommendation process. So, our approach only produces minimal overhead to service times.

References

[1] J. Andersson, A. Heberle, J. Kirchner, and W. Löwe, "Service Level Achievements - Distributed knowledge for optimal service selection," in Ninth IEEE European Conference on Web Services (ECOWS), 2011, pp. 125–132.

[2] C. Sammut and M. Harries, "Concept drift," in Encyclopedia of Machine Learning, C. Sammut and G. Webb, Eds. Springer US, 2010, pp. 202–205. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-30164-8_153

[3] C. Shearer, "The CRISP-DM model: The new blueprint for data mining," Journal of Data Warehousing, vol. 5, no. 4, 2000, pp. 13–22.

[4] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," Informatica, no. 31, 2007, pp. 249–268.

[5] J. Han and M. Kamber, Data Mining: Concepts and Techniques, 2nd ed. Elsevier, Morgan Kaufmann, 2006.

[6] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2001, pp. 97–106.

[7] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," Data Mining and Knowledge Discovery, vol. 23, no. 1, 2011, pp. 128–168.

[8] E. J. Keogh and M. J. Pazzani, "Learning augmented bayesian classifiers: A comparison of distribution-based and classification-based approaches," 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.7726

[9] Weka, "Weka Javadoc – Hoeffding Tree," date of retrieval: 25 Oct 2014; http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/HoeffdingTree.html.

[10] KDnuggets.com – Data Mining Community's Top Resource for Data Mining and Analytics Software, "What analytics, data mining, data science software/tools you used in the past 12 months for a real project poll," June 2014, date of retrieval: 22 Oct 2014; http://www.kdnuggets.com/polls/2014/analytics-data-mining-data-science-software-used.html.

[11] "RapidMiner," https://rapidminer.com/, http://sourceforge.net/projects/rapidminer/.

[12] "The R project for Statistical Computing," http://www.r-project.org/.

[13] "Apache Mahout," http://mahout.apache.org/.

[14] "Apache Spark," http://spark.apache.org/.

[15] "KNIME," http://www.knime.org/.

[16] "The SHOGUN Machine Learning Toolbox," http://www.shogun-toolbox.org/.

[17] "Shark machine learning library," http://image.diku.dk/shark/sphinx_pages/build/html/index.html.

[18] "scikit-learn – Machine Learning in Python," http://scikit-learn.org/.

[19] "Vowpal Wabbit (Fast Learning)," http://hunch.net/~vw/.

[20] Machine Learning Group at the University of Waikato, "Weka – Data mining with open source machine learning software in Java," http://www.cs.waikato.ac.nz/ml/weka/.

[21] University of Waikato, "MOA Massive Online Analysis," http://moa.cms.waikato.ac.nz/.

[22] P. Karg, "Evaluation and Implementation of Machine Learning Methods for an Optimized Web Service Selection in a Future Service Market," Master's thesis, Karlsruhe University of Applied Sciences/Linnaeus University, Germany/Schweden, 2014.

[23] Z. Zheng, H. Ma, M. Lyu, and I. King, "QoS-aware Web service recommendation by collaborative filtering," Services Computing, IEEE Transactions on, vol. 4, no. 2, 2011, pp. 140–152.

[24] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for QoS-based service recommendation," in Web Services (ICWS), IEEE 19th International Conference on, 2012, pp. 202–209.

[25] L. Kuang, Y. Xia, and Y. Mao, "Personalized services recommendation based on context-aware QoS prediction," in Web Services (ICWS), IEEE 19th International Conference on, 2012, pp. 400–406.

[26] R. Yang, Q. Chen, L. Qi, and W. Dou, "A QoS evaluation method for personalized service requests," in Web Information Systems and Mining, ser. Lecture Notes in Computer Science, vol. 6988. Springer Heidelberg, 2011, pp. 393–402.

[27] G. Kang, J. Liu, M. Tang, X. Liu, B. Cao, and Y. Xu, "AWSR: Active Web service recommendation based on usage history," in Web Services (ICWS), IEEE 19th International Conference on, 2012, pp. 186–193.

[28] Q. Yu, "Decision tree learning from incomplete QoS to bootstrap service recommendation," in Web Services (ICWS), IEEE 19th International Conference on, 2012, pp. 194–201.

[29] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, and F. Leymann, "Runtime prediction of service level agreement violations for composite services," in Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops, 2010, pp. 176–186.